链滴

docker 快速部署 lnmp 开发环境

作者：450370050

原文链接：https://ld246.com/article/1524193297454

来源网站：链滴

Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器。我们通过部署lnmp环境，制作php镜像，来熟悉怎么快速的利用docker部署开发环境。

nginx部署

1.创建服务运行网络

```
docker network create -d bridge my-net
```

2.nginx镜像下载

```
docker pull hub.c.163.com/library/nginx:latest
```

3.编写nginx配置

```
mkdir -p /home/nginx/ && cd /home/nginx/ && vi nginx.conf
```

```
user  nginx;
worker_processes  1;
error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

events {
    worker_connections  1024;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
    '$status $body_bytes_sent "$http_referer" '
    '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;
    sendfile        on;
    keepalive_timeout  65;
    events {
        worker_connections  1024;  ## Default: 1024
    }
    server {
        listen       80;
        server_name  localhost;

            location / {
                root   /usr/share/nginx/html;
                index  index.html index.htm;
            }
            error_page   500 502 503 504  /50x.html;
            location = /50x.html {
```

```
            root   /usr/share/nginx/html;
        }
    }
    include /etc/nginx/conf.d/*.conf;
}
```

4.启动nginx镜像，创建容器

```
docker run -p 80:80 -v /home/nginx/nginx.conf:/etc/nginx/nginx.conf -v /home/nginx/cond.d
:/etc/nginx/conf.d  -v /home/nginx/log/:/var/log/nginx/ -v /home/www/:/usr/share/nginx/ht
l/ --name my-nginx  --network my-net -ti  hub.c.163.com/library/nginx:latest
```

-v 挂在目录

--network 指定我们nginx容器运行的网络(<font color=red>同一个网络的容器可以互相访问</fon
>)

```
###我们看到 nginx容器已经启动      0.0.0.0:80->80  80端口映射  我通过宿主机的80端口访问到容器

[root@localhost php-fpm]# docker ps -a
CONTAINER ID        IMAGE                                COMMAND                  CREATED             STATU
            PORTS               NAMES
8c134ba045bb        hub.c.163.com/library/nginx:latest   "nginx -g 'daemon ..."   3 seconds ago
    Up 2 seconds        0.0.0.0:80->80/tcp   my-nginx
```

mysql部署

1.mysql镜像下载

```
docker pull hub.c.163.com/library/mysql:latest
```

2.编写mysql配置

```
mkdir -p /home/mysql/ && cd /home/mysql/ && vi my.cnf

[mysql]
pid-file    = /var/run/mysqld/mysqld.pid
socket      = /var/run/mysqld/mysqld.sock
datadir      = /var/lib/mysql
symbolic-links=0

[mysqld]
skip-host-cache
skip-name-resolve
```

3.启动mysql镜像，创建容器

```
docker run -v /home/mysql/data/:/var/lib/mysql/  --name my-mysql -e MYSQL_ROOT_PASS
ORD=12345678 -p 3306:3306  -tid --network my-net hub.c.163.com/library/mysql:latest

##我们看到 mysql容器已经启动   也可以通过navicat等客户端工具连接访问
```

```
[root@localhost log]# docker ps -a
CONTAINER ID        IMAGE                                COMMAND                  CREATED             STAT
S                   PORTS                    NAMES
22aaa2e60073        hub.c.163.com/library/mysql:latest   "docker-entrypoint..."   About a minut
 ago   Up About a minute          0.0.0.0:3306->3306/tcp   my-mysql
8c134ba045bb        hub.c.163.com/library/nginx:latest   "nginx -g 'daemon ..."   52 seconds ag
      Up 2 seconds        0.0.0.0:80->80/tcp         my-nginx
```

php部署

1.php镜像下载

```
docker pull hub.c.163.com/library/php:7.2.0RC1-fpm
```

2.编写php配置

```
mkdir -p /home/php/ && cd /home/php/ && vi php-fpm.conf
```

```
[global]
error_log = /var/log/php-fpm/error.log
daemonize = no

[www]
access.log =  /var/log/php-fpm/access.log
clear_env = no
catch_workers_output = yes
user = www-data
group = www-data
listen = 127.0.0.1:9000
pm = dynamic
pm.max_children = 5
pm.start_servers = 2
pm.min_spare_servers = 1
pm.max_spare_servers = 3
listen = [::]:9000
```

3.启动php镜像，创建容器

```
docker run  -v /home/php-fpm/php-fpm.conf:/usr/local/etc/php-fpm.conf -v /home/php-fpm
log/:/var/log/php-fpm/ -v /home/www/:/home/www/  --name my-php-fpm --network my-ne
  -tid  hub.c.163.com/library/php:7.2.0RC1-fpm
```

启动php-fpm镜像时并没有映射端口到宿主机，那怎么访问呢。

我们可以通过nginx的反向，访问php-fpm，这样我们的php-fpm就可以通过nginx正常访问php了。

```
####my-php-fpm   我们php-fpm的容器名
location ~ \.php$ {
    fastcgi_pass   my-php-fpm:9000;
    fastcgi_index  index.php;
    fastcgi_param  SCRIPT_FILENAME  /home/www/$fastcgi_script_name;
```

```
    include        fastcgi_params;
}

##现在我们可以看到容器服务都正常启动了
[root@localhost www]# docker ps -a
CONTAINER ID        IMAGE                                COMMAND                  CREATED             STA
US              PORTS                    NAMES
4605179a5653        hub.c.163.com/library/php:7.2.0RC1-fpm   "docker-php-entryp..."   2 minu
es ago      Up 2 minutes        9000/tcp                 my-php-fpm
22aaa2e60073        hub.c.163.com/library/mysql:latest       "docker-entrypoint..."   15 minutes
go      Up 15 minutes       0.0.0.0:3306->3306/tcp   my-mysql
8c134ba045bb        hub.c.163.com/library/nginx:latest       "nginx -g 'daemon ..."   24 minutes
go      Up About a minute   0.0.0.0:80->80/tcp       my-nginx
```

测试服务

curl访问测试的文件

```
[root@localhost www]# echo '<?php echo "hello docker lnmp server"; ' > /home/www/index.
hp
[root@localhost www]# curl 192.168.139.136/index.php
hello docker lnmp server
[root@localhost www]#
```

制作自己的镜像

服务已经运行，我们的php要使用mysqli、redis等扩展，怎么办？docker的php镜像已经有了添加展的工具docker-php-ext-*，我们可以通过工具制作新的镜像使用。

下面我们制作一个支持 gd、mysqli、iconv的镜像

1. 编写Dockerfile文件

```
Vi Dockerfile


##基础镜像
FROM hub.c.163.com/library/php:7.2.0RC1-fpm

RUN curl -k  'https://ftp.acc.umu.se/mirror/gnu.org/savannah/freetype/freetype-2.4.0.tar.gz' -
 freetype.tar.gz  \
    && tar -zxvf freetype.tar.gz \
    && rm  freetype.tar.gz \
    && cd freetype-2.4.0 \
    && ./configure \
    && make -j \
    && make install && cd .. && rm -rf freetype-2.4.0

RUN pecl install igbinary  \
    && docker-php-ext-enable igbinary

RUN pecl install redis-3.1.5 \
```

```
    && docker-php-ext-enable redis

RUN  curl -k 'http://219.238.7.71/files/4239000009B52EEA/www.zlib.net/zlib-1.2.11.tar.gz'  -o lib.tar.gz \
    && tar -zxvf zlib.tar.gz  \
    && rm  zlib.tar.gz \
    && cd zlib-1.2.11 \
    && ./configure  \
    && make -j  \
    && make install && cd .. && rm -rf zlib-1.2.11 \
    && curl -k  'https://ftp-osl.osuosl.org/pub/libpng/src/libpng16/libpng-1.6.34.tar.gz' -o libpg.tar.gz \
    && tar -zxvf libpng.tar.gz \
    && rm  libpng.tar.gz \
    && cd libpng-1.6.34 \
    && ./configure \
    && make -j \
    && make install && cd .. && rm -rf libpng-1.6.34

RUN docker-php-ext-install mysqli  pdo_mysql  iconv gd
```

2.构建我们的镜像

```
docker build -t php-fpm-ext .

## 我们看到镜像列表中已经创建出  php-fpm-ext镜像

[root@localhost php-fpm]# docker  images
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
php-fpm-ext                 latest              818c2a9a9b49        6 seconds ago       385MB
hub.c.163.com/library/php   7.2.0RC1-fpm        96a7430411ac        3 months ago        368MB
hub.c.163.com/library/nginx latest              46102226f2fd        7 months ago        109MB
hub.c.163.com/library/mysql latest              9e64176cd8a2        8 months ago        407MB
```

删除之前的my-php-fpm容器，用我们制作的镜像启动php-fpm容器，看下phpinfo中是否已经存在加的扩展。